

Spring Boot Course Project Documentation

Overview

This Spring Boot project is designed as a demo for managing course-related information. It includes RESTful endpoints to perform CRUD operations on courses and is integrated with a MySQL database for data persistence. The project also includes basic configurations and dependencies required to run a Spring Boot application.

Project Structure

```
```\nsrc/main/java\n|-- com/springrest\n|   |-- controller\n|       |-- MyController.java\n|   |-- dao\n|       |-- CourseDao.java\n|   |-- entities\n|       |-- Course.java\n|   |-- services\n|       |-- CourseService.java\n|       |-- CourseServiceImpl.java\n|-- SpringrestApplication.java\nsrc/main/resources\n|-- application.properties\npom.xml\n```\n
```

## Maven Configuration

`pom.xml`

The `pom.xml` file manages project dependencies and plugins. Key dependencies include:

- **\*\*Spring Boot Starter Data JPA\*\***: For integrating JPA with the Spring Boot application.
- **\*\*Spring Boot Starter Web\*\***: To build web, including RESTful, applications using Spring MVC.
- **\*\*MySQL Connector Java\*\***: For connecting the application to a MySQL database.
- **\*\*Springdoc OpenAPI\*\***: To generate API documentation.
- **\*\*Spring Boot Starter Test\*\***: For testing the application.

- **\*\*Spring Boot Starter Actuator\*\***: For adding production-ready features to help monitor and manage the application.

```
```.xml
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.5.0</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>
</dependencies>
```.
```

### Application Properties

#### `application.properties`

This file contains the configuration for connecting to the MySQL database.

```
```.properties
spring.datasource.url=jdbc:mysql://localhost:3306/yourdatabase
spring.datasource.username=yourusername
```

```
spring.datasource.password=yourpassword
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```
```

### Application Entry Point

#### `SpringrestApplication.java`

The main entry point for the Spring Boot application.

```
```java
package com.springrest;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringrestApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringrestApplication.class, args);
    }

}
```
```

## Controllers

`MyController.java`

This controller handles HTTP requests related to courses.

```
```java
package com.springrest.controller;

import com.springrest.entities.Course;
import com.springrest.services.CourseService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

```

@RestController
@RequestMapping("/api/courses")
public class MyController {

    @Autowired
    private CourseService courseService;

    @GetMapping
    public List<Course> getCourses() {
        return courseService.getCourses();
    }

    @GetMapping("/{courseId}")
    public Course getCourse(@PathVariable Long courseId) {
        return courseService.getCourse(courseId);
    }

    @PostMapping
    public Course addCourse(@RequestBody Course course) {
        return courseService.addCourse(course);
    }

    @PutMapping("/{courseId}")
    public Course updateCourse(@RequestBody Course course, @PathVariable Long
courseId) {
        return courseService.updateCourse(course, courseId);
    }

    @DeleteMapping("/{courseId}")
    public void deleteCourse(@PathVariable Long courseId) {
        courseService.deleteCourse(courseId);
    }
}

```

Services

`CourseService.java`

Interface for course services.

```

```java
package com.springrest.services;

```

```
import com.springrest.entities.Course;
```

```
import java.util.List;
```

```
public interface CourseService {
 List<Course> getCourses();
 Course getCourse(Long courseId);
 Course addCourse(Course course);
 Course updateCourse(Course course, Long courseId);
 void deleteCourse(Long courseId);
}
```
```

```
#### `CourseServiceImpl.java`
```

Implementation of the `CourseService` interface.

```
` `` `java
```

```
package com.springrest.services;
```

```
import com.springrest.dao.CourseDao;  
import com.springrest.entities.Course;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

```
@Service
```

```
public class CourseServiceImpl implements CourseService {
```

```
    @Autowired
```

```
    private CourseDao courseDao;
```

```
    @Override
```

```
    public List<Course> getCourses() {  
        return courseDao.findAll();  
    }
```

```
    @Override
```

```
    public Course getCourse(Long courseId) {  
        return courseDao.findById(courseId).orElse(null);  
    }
```

```
    @Override
```

```

public Course addCourse(Course course) {
    return courseDao.save(course);
}

@Override
public Course updateCourse(Course course, Long courseId) {
    Course existingCourse = courseDao.findById(courseId).orElse(null);
    if (existingCourse != null) {
        existingCourse.setTitle(course.getTitle());
        existingCourse.setDescription(course.getDescription());
        return courseDao.save(existingCourse);
    }
    return null;
}

@Override
public void deleteCourse(Long courseId) {
    courseDao.deleteById(courseId);
}
}
```

```

## DAO (Data Access Object)

#### `CourseDao.java`

Repository interface for course entities.

```

```java
package com.springrest.dao;

import com.springrest.entities.Course;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CourseDao extends JpaRepository<Course, Long> {
}
```

```

## Entities

#### `Course.java`

Entity class representing the Course.

```

` `` `java
package com.springrest.entities;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Course {

 @Id
 @GeneratedValue(strategy = GenerationType.IDENTITY)
 private Long id;
 private String title;
 private String description;

 // Getters and Setters
}
` `` `

```

## Running the Application

1. **\*\*Set Up MySQL Database\*\***: Ensure you have a MySQL database set up and configured as per the `application.properties` file.
2. **\*\*Build the Project\*\***: Run `mvn clean install` to build the project.
3. **\*\*Run the Application\*\***: Use `mvn spring-boot:run` to start the application.
4. **\*\*Access API\*\***: The API can be accessed at `http://localhost:8080/api/courses`.

## API Endpoints

- **\*\*GET /api/courses\*\***: Retrieve all courses.
- **\*\*GET /api/courses/{courseId}\*\***: Retrieve a specific course by ID.
- **\*\*POST /api/courses\*\***: Add a new course.
- **\*\*PUT /api/courses/{courseId}\*\***: Update an existing course.
- **\*\*DELETE /api/courses/{courseId}\*\***: Delete a course by ID.